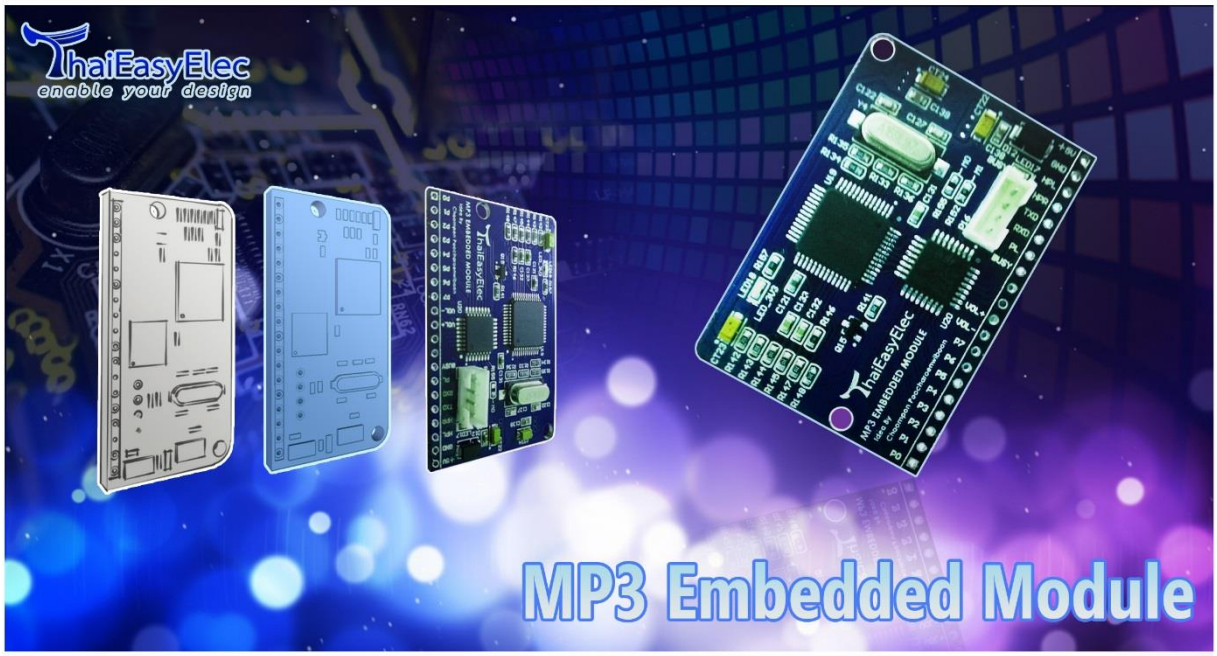
User Manual

ThaiEasyElec MP3 Embedded Module

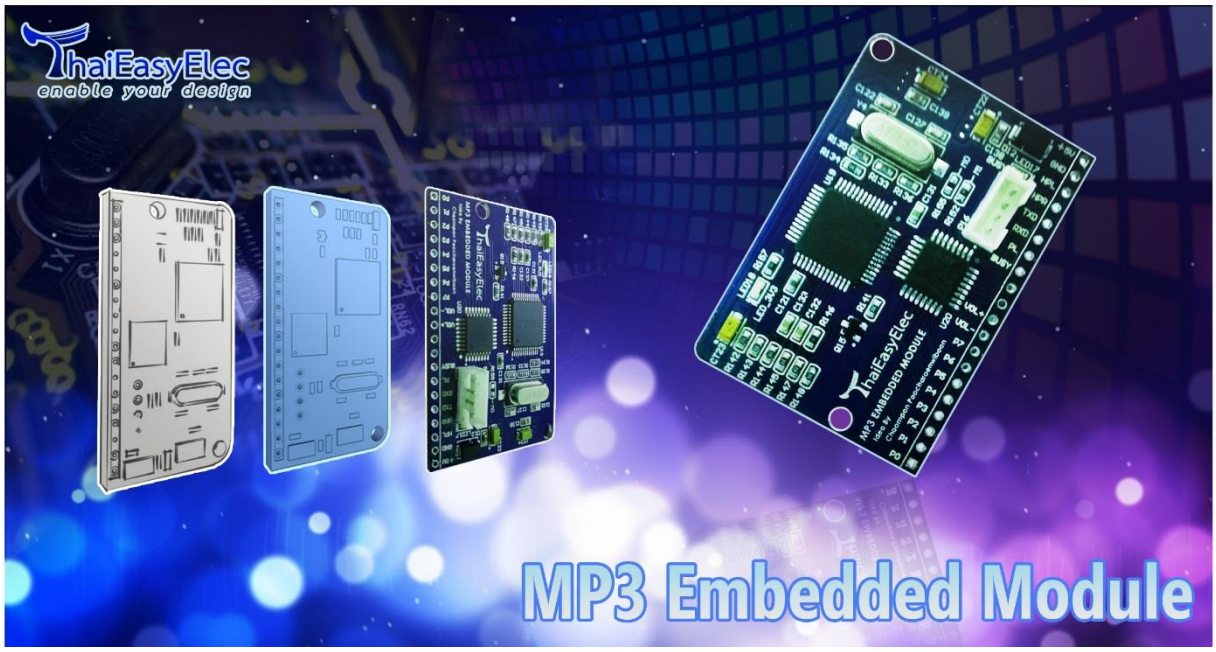
Version 1.2



Thaieasyelec
enable your design

Revision History

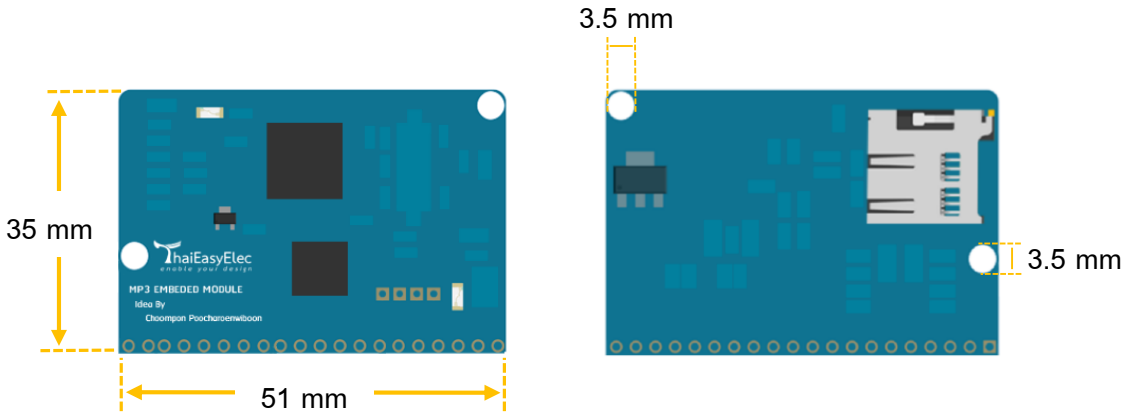
Version	Date	Changes
1.2	4 June 2014	Wording Correction
1.1	16 November 2013	Image chaged, Usage with Arduino added
1.0	3 October 2013	Original Version



The MP3 Module (ETEE043) was designed to be an easy to use, cost-effective MP3 playback module. It uses BU94502AKS2 as the MP3 decoder which provides fine quality sound with 8 kHz – 48 kHz sampling rate. The user can use any MCU to control the module using parallel or serial interface. With MP3 files stored on a SD card, the user can select a sound folder, select a file, start playback and control play volume using simple serial or parallel commands. The SD card can be as large as 32 GByte. The module is well suited many applications such as queuing system, voice mail box, station announcement.

Features

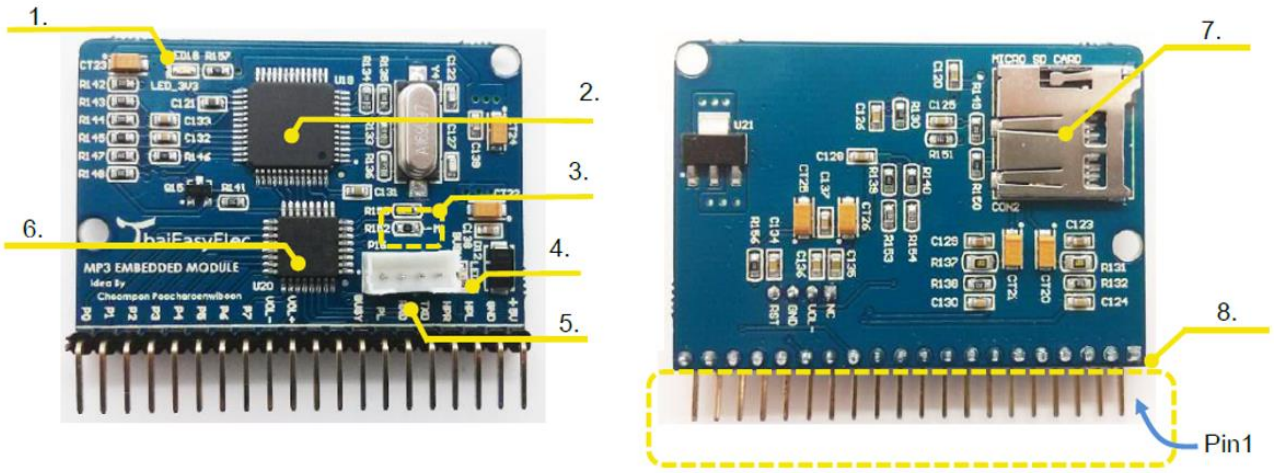
- Support stereo MP3 files
- 2 interface modes: serial and parallel
- Support FAT32 and FAT16 file system
- Accept micro SD Card with SDHC supported (maximum 32 GB)
- Sampling rates: 8 kHz - 48 kHz.
- Accept 5 VDC power supply
- Use 20-pin 2.54mm-pitch header for interface
- Dimension: 35x51 mm



Electrical Characteristic

Parameter	Min.	Typ.	Max.	Unit.
Operating voltage	4	5	9	V
Operating current	-	-	150	mA
Input voltage	3.1	-	5.3	V
Output voltage	3.1	-	-	V
Audio Distortion rate	-	0.005	0.05	%
Audio Dynamic range	-	90	-	dB
Audio Max output level	0.6	0.75	-	Vrms

Board Description



No. 1 5V power indicator

N0. 2 MP3 Decoder IC, BU94502AKS2

No. 3 Mode Setting (Serial / Parallel) for soldering

No. 4 BUSY LED

No. 5 MCU Debug Port

No. 6 MCU, STM8S103

No. 7 Micro-SD Card Socket

No. 8 Interface Header, see detail below:

Pin No.	Pin Name	Description
1 - 8	P0 – P7	Data Input
9	VOL-	Volume down
10	VOL+	Volume up
11	RESERVE	-
12	RESERVE	-
13	BUSY	Busy, low active
14	PL	Trigger Input, low active
15	RXD	Serial port receiver input
16	TXD	Serial port transmitter output
17	HPR	Audio output right
18	HPL	Audio output left
19	GND	Ground
20	+5V	5V Power input

Note!!

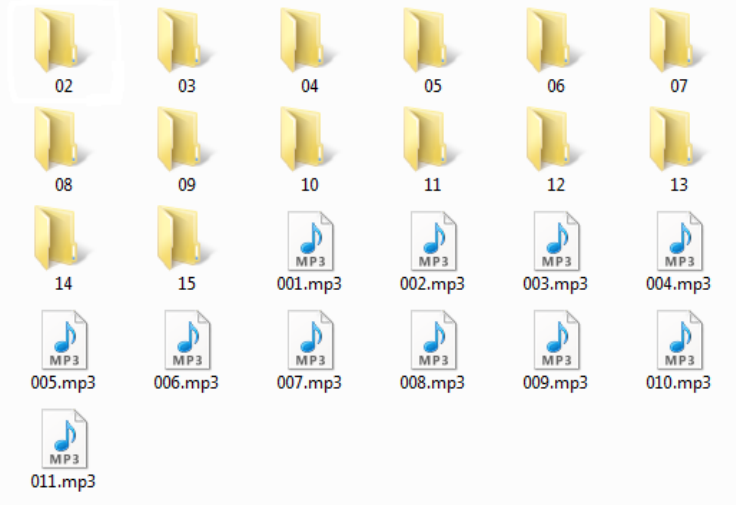
BUSY Pin: BUSY is driven low on initialization state and when a file is being played. In all other case, the pin is driven high.

PL Pin: PL is the trigger pin with low-level active. When asserted, the module read data from P0-P7. It must be asserted when data on P0-P7 is ready.

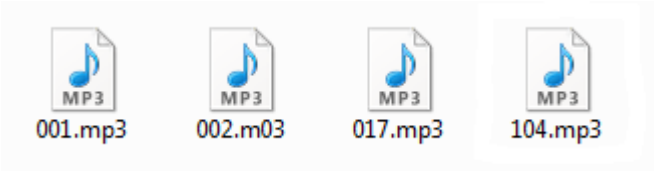
How to use the module

1. Prepare files on SD card

Sound files and folders have to be named sequentially with right format so the module is able to get to the next file or folder. The file names in each folder must be 001.mp3, 002.mp3 ... 199.mp3. So there can be 199 files in a folder. And there can be 15 folders on a SD card with names: 02, 03 ... 15. The root directory is considered as the folder 1.

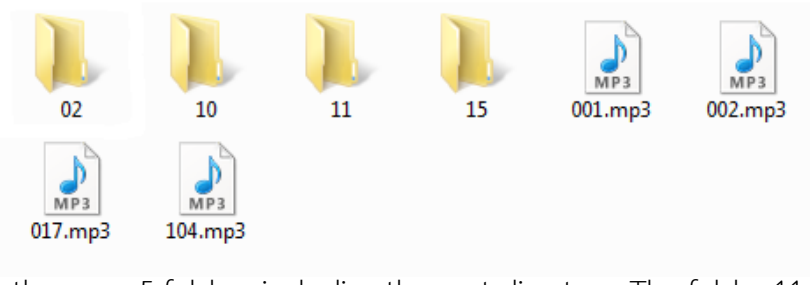


- File ordering example



From figure above, there are 4 files and the file 017.mp3 is counted as the 3rd file. So the user has to play it using command 0x03.

- Folder ordering example



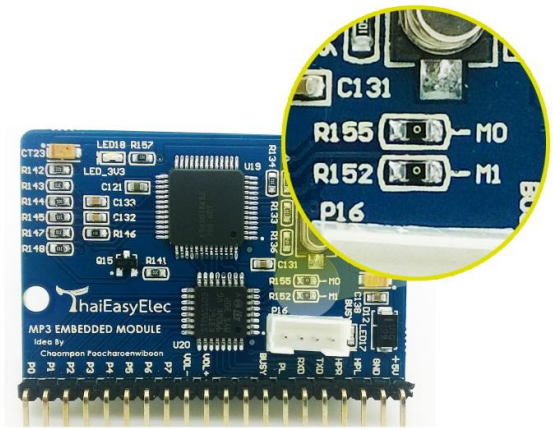
From figure above, there are 5 folders including the root directory. The folder 11 is counted as the 4th folder, to get into it, use command 0xF4 (Change to folder 4).

As illustrated, it's recommended to name all files and folders sequentially.

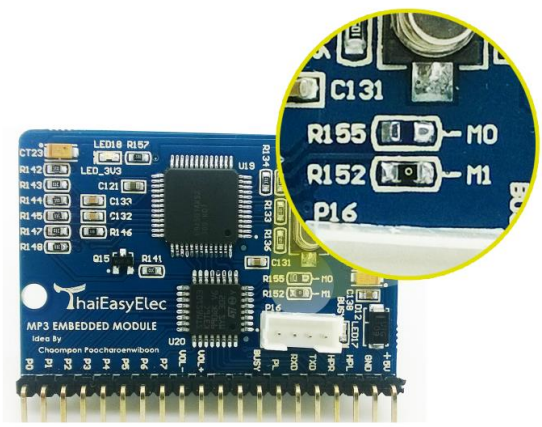
2. Select the operating mode (Serial / Parallel)

2.1 To use Parallel mode, M0 and M1 must be connected to GND as shown below (zero-ohm resistors are used here):

*** By default, the module is configured in parallel mode ***

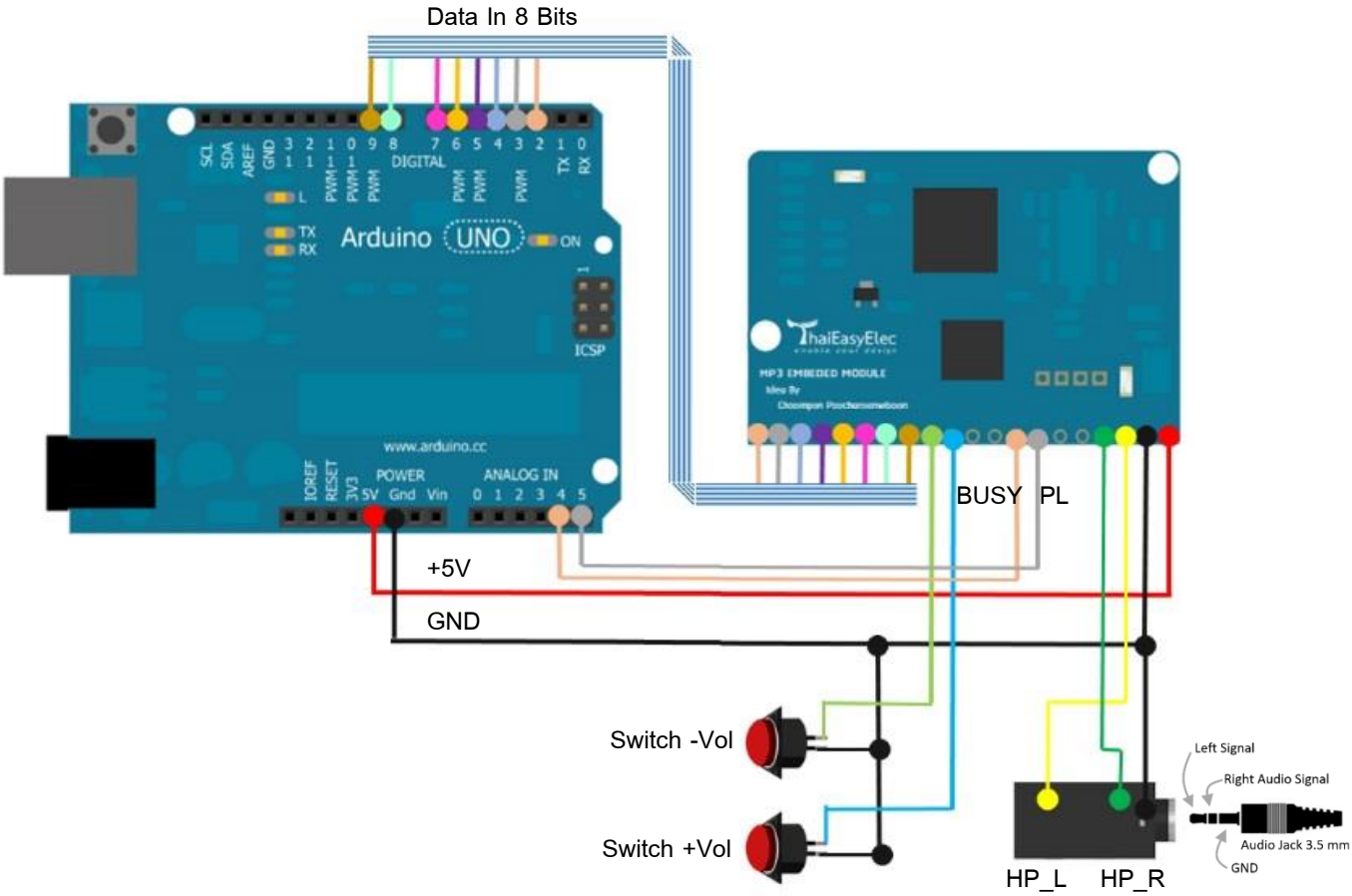


2.2 To use Serial mode, remove M0 and leave M1 connected to GND. See image below:



3. Control the module in Parallel mode

The module is controlled by Arduino UNO R3 (EADN014) for example. In parallel mode, 10 signals are used for control: 8-bit data, PL and BUSY. Optionally, +VOL and –VOL can be connected to switches for increasing or decreasing the volume directly.



The connections are described below:

MP3 module	Pin	Device
+5V	+5V	Arduino
GND	GND	Arduino
P0–P7	D2-D9	Arduino
BUSY	A4	Arduino
PL	A5	Arduino
VOL+	-	Switch 1
VOL-	-	Switch 2
HP_L	Left	Audio Jack
HP_R	Right	Audio Jack

From the circuit above, control signals are P0-P7, PL and BUSY. Whenever that BUSY is “High”, the module is ready to accept new parallel command. The user can send a command to P0-P7 then send a “Low” strobe pulse on PL. Commands are listed below:

Command Word										Function
BIN								HEX	DEC	
P7	P6	P5	P4	P3	P2	P1	P0			
0	0	0	0	0	0	0	1	0x01	001	Play the first file
0	0	0	0	0	0	1	0	0x02	002	Play the second file
…	….	…	…	…	….	….	…	…	…..	….
1	1	0	0	0	1	1	1	0xC7	199	Play the 199th file
1	1	0	0	1	0	0	0	0xC8	200	Set volume level to 0 (the lowest level)
1	1	0	0	1	0	0	1	0xC9	201	Set volume level to 1
…	….	…	…	…	….	….	…	…	…..	….
1	1	1	0	0	1	1	1	0xE7	231	Set volume level to 31 (the highest level)
1	1	1	0	1	0	0	0	0xE8	232	Increase volume by 1 level
1	1	1	0	1	0	0	1	0xE9	233	Decrease volume by 1 level
1	1	1	0	1	0	1	1	0xEB	235	Pause playback
1	1	1	0	1	1	0	0	0xEC	236	Resume playback
1	1	1	0	1	1	1	1	0xEF	239	Stop playback
1	1	1	1	0	0	0	1	0xF1	241	Change directory to 01 (Root directory)
1	1	1	1	0	0	1	0	0xF2	242	Change directory to 02
…	….	…	…	…	….	….	…	…	…..	….
1	1	1	1	1	1	1	1	0xFF	255	Change directory to 15

Example 1 In parallel mode, send a command to play 001.mp3.

```
const int P0 = 2;
const int P1 = 3;
const int P2 = 4;
const int P3 = 5;
const int P4 = 6;
const int P5 = 7;
const int P6 = 8;
const int P7 = 9;
const int BUSY = A4;
const int PL = A5;

void set_port() {
   pinMode(P0, OUTPUT);
   pinMode(P1, OUTPUT);
   pinMode(P2, OUTPUT);
   pinMode(P3, OUTPUT);
   pinMode(P4, OUTPUT);
   pinMode(P5, OUTPUT);
   pinMode(P6, OUTPUT);
   pinMode(P7, OUTPUT);
   pinMode(PL, OUTPUT);
   pinMode(BUSY,INPUT);
}

boolean cmd_stop =false;

void setup()  {
   set_port();
}

void loop() {

   if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)){
    digitalWrite(P0,HIGH);
    digitalWrite(P1,LOW);
    digitalWrite(P2,LOW);
    digitalWrite(P3,LOW);
    digitalWrite(P4,LOW);
    digitalWrite(P5,LOW);
    digitalWrite(P6,LOW);
    digitalWrite(P7,LOW);

    delay(100);
    digitalWrite(PL, LOW);
    delay(100);
    digitalWrite(PL, HIGH);


    cmd_stop = true;
  }
}
```

1. Pin assignments

2. Pins' direction assignments

3. Call set_port() to set pins' direction

4. When BUSY is "High", set data pins to 0x01 to the module and send "Low"

Code description:

```
void set_port() {
   pinMode(P0, OUTPUT);
   pinMode(P1, OUTPUT);
   pinMode(P2, OUTPUT);
   pinMode(P3, OUTPUT);
   pinMode(P4, OUTPUT);
   pinMode(P5, OUTPUT);
   pinMode(P6, OUTPUT);
   pinMode(P7, OUTPUT);
   pinMode(PL, OUTPUT);
   pinMode(BUSY,INPUT);
}
```

Function set_port () is used to initialized pins to be output or input.

```
void setup()  {
  set_port();

}
```

Function setup() is used for initialization.

```
void loop() {
  if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)){
    digitalWrite(P0,HIGH);
    digitalWrite(P1,LOW);
    digitalWrite(P2,LOW);
    digitalWrite(P3,LOW);
    digitalWrite(P4,LOW);
    digitalWrite(P5,LOW);
    digitalWrite(P6,LOW);
    digitalWrite(P7,LOW);

    delay(100);
    digitalWrite(PL, LOW);
    delay(100);
    digitalWrite(PL, HIGH);

    cmd_stop = true;
  }
}
```

When logic on BUSY pin becomes “High” and cmd_stop is false, a command 0x01 will be sent by setting 8-bit data to 00000001 then send a “Low” strobe pulse on PL. Then cmd_stop is set to true. This will command the module to play the first file once.

Example 2 In parallel mode, send a command to play 002.mp3 in folder 03.

```
const int P0 = 2;
const int P1 = 3;
const int P2 = 4;
const int P3 = 5;
const int P4 = 6;
const int P5 = 7;
const int P6 = 8;
const int P7 = 9;
const int BUSY = A4;
const int PL = A5;
```

1. Pin assignments

```
void set_port() {
  pinMode(P0, OUTPUT);
  pinMode(P1, OUTPUT);
  pinMode(P2, OUTPUT);
  pinMode(P3, OUTPUT);
  pinMode(P4, OUTPUT);
  pinMode(P5, OUTPUT);
  pinMode(P6, OUTPUT);
  pinMode(P7, OUTPUT);
  pinMode(PL, OUTPUT);
  pinMode(BUSY,INPUT);
}
```

2. Pins' direction assignments

```
void send_CMD(unsigned char cmd) {
  if( cmd&0x01 ) digitalWrite(P0, HIGH); //Bit 0
    else digitalWrite(P0, LOW);
  if( cmd&0x02 ) digitalWrite(P1, HIGH); //Bit 1
    else digitalWrite(P1, LOW);
  if( cmd&0x04 ) digitalWrite(P2, HIGH); //Bit 2
    else digitalWrite(P2, LOW);
  if( cmd&0x08 ) digitalWrite(P3, HIGH); //Bit 3
    else digitalWrite(P3, LOW);
  if( cmd&0x10 ) digitalWrite(P4, HIGH); //Bit 4
    else digitalWrite(P4, LOW);
  if( cmd&0x20 ) digitalWrite(P5, HIGH); //Bit 5
    else digitalWrite(P5, LOW);
  if( cmd&0x40 ) digitalWrite(P6, HIGH);  //Bit 6
    else digitalWrite(P6, LOW);
  if( cmd&0x80 ) digitalWrite(P7, HIGH);  //Bit 7
    else digitalWrite(P7, LOW);

  delay(100); digitalWrite(PL, LOW);
  delay(100); digitalWrite(PL, HIGH);
}
```

3. Set data pins according to sending command

```
boolean cmd_stop =false ;

void setup()  {
  set_port();
}
```

4. Call set_port() to set pins' direction

```
void loop()  {
  if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)) {
     send_CMD(0xF3);
     delay(200);
       while(digitalRead(BUSY) == HIGH) {
         send_CMD(0x02);
         cmd_stop=true;
       }
    }
}
```

5. When BUSY is "High", send 0xF3 and 0x02 to the module. To change directory to 03 and play file 002.mp3.

Code description:

```
void send_CMD(unsigned char cmd) {
  if( cmd&0x01 ) digitalWrite(P0, HIGH); //Bit 0
    else digitalWrite(P0, LOW);
  if( cmd&0x02 ) digitalWrite(P1, HIGH); //Bit 1
    else digitalWrite(P1, LOW);
  if( cmd&0x04 ) digitalWrite(P2, HIGH); //Bit 2
    else digitalWrite(P2, LOW);
  if( cmd&0x08 ) digitalWrite(P3, HIGH); //Bit 3
    else digitalWrite(P3, LOW);
  if( cmd&0x10 ) digitalWrite(P4, HIGH); //Bit 4
    else digitalWrite(P4, LOW);
  if( cmd&0x20 ) digitalWrite(P5, HIGH); //Bit 5
    else digitalWrite(P5, LOW);
  if( cmd&0x40 ) digitalWrite(P6, HIGH);  //Bit 6
    else digitalWrite(P6, LOW);
  if( cmd&0x80 ) digitalWrite(P7, HIGH);  //Bit 7
    else digitalWrite(P7, LOW);

  delay(100); digitalWrite(PL, LOW);
  delay(100); digitalWrite(PL, HIGH);
}
```

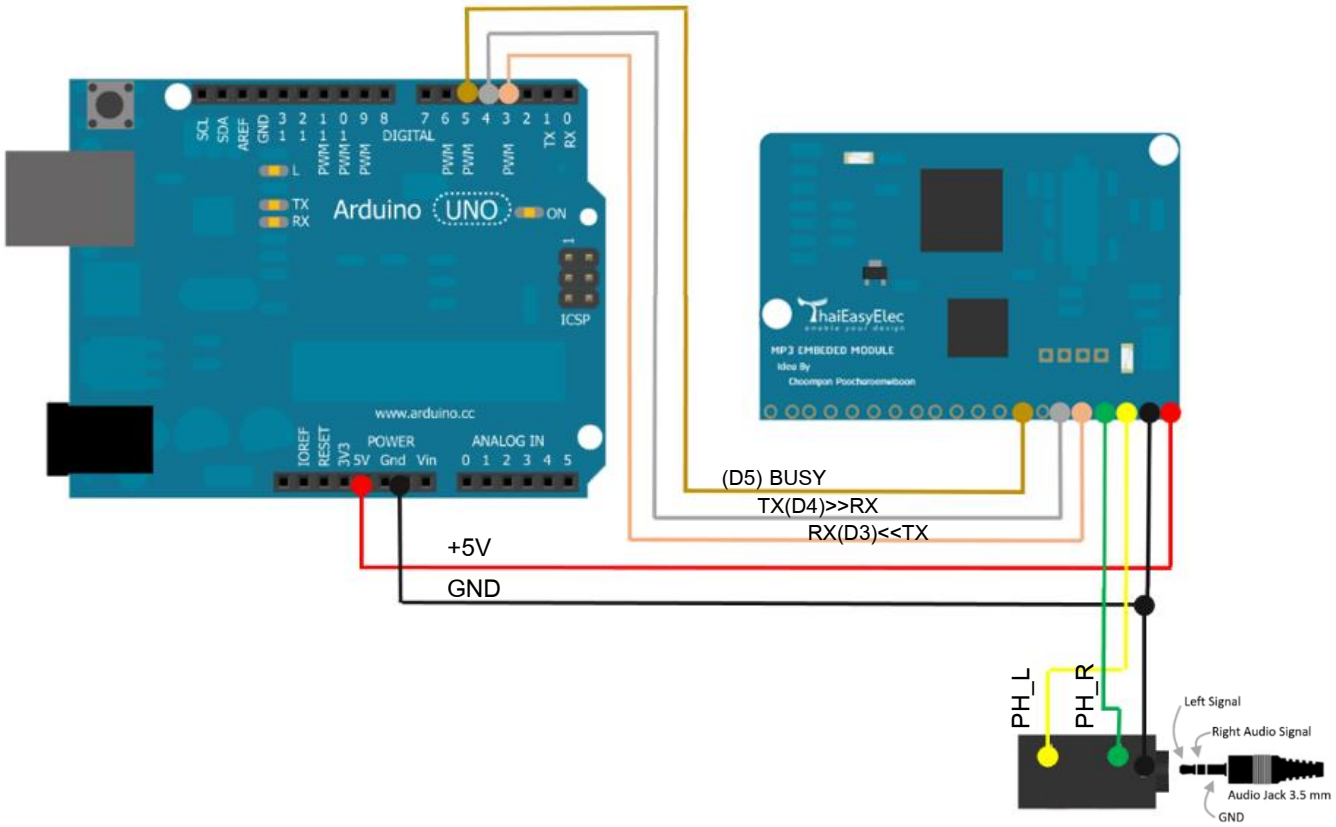
Function send_CMD(unsigned char cmd) wraps the code in the first example into a single function. It accepts a character named cmd, checks cmd bit-by-bit and set 8-bit data pins according to cmd. Then it sends a strobe pulse on PL.

```
void loop()  {
 if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)) {
   send_CMD(0xF3);
   delay(200);
      while(digitalRead(BUSY) == HIGH) {
       send_CMD(0x02);
       cmd_stop=true;
      }
   }
}
```

When logic on BUSY pin becomes “High” and cmd_stop is false, a command 0xF3 will be sent to change directory to 03. Then wait until BUSY becomes “High” again and a command 0x02 is sent to play 002.mp3.

4. Control the module in Serial mode

Circuit below shows how to connect the module to Arduino UNO R3 (EADN014) using 3 signals: TX, RX and BUSY.



MP3 module	Pin	Device
+5V	+5V	Arduino
GND	GND	Arduino
TX	D4	Arduino
RX	D3	Arduino
BUSY	D5	Arduino
VOL+	-	Switch 1
VOL-	-	Switch 2
HP_L	Left	Audio Jack
HP_R	Right	Audio Jack

In this example, a software serial is used by setting D3 to be RX and D4 to be TX. The communication is configured to:

Baud rate: 9600, Data Bit : 8, Stop Bit :1, Parity : None

Commands are exactly the same as used in parallel mode:

Command word		Function
HEX	DEC	
0x01	001	Play the first file
0x02	002	Play the second file
…	…..	….
0xC7	199	Play the 199th file
0xC8	200	Set volume level to 0 (the lowest level)
0xC9	201	Set volume level to 1
…	…..	….
0xE7	231	Set volume level to 31 (the highest level)
0xE8	232	Increase volume by 1 level
0xE9	233	Decrease volume by 1 level
0xEB	235	Pause playback
0xEC	236	Resume playback
0xEF	239	Stop playback
0xF1	241	Change directory to 01 (Root directory)
0xF2	242	Change directory to 02
…	…..	….
0xFF	255	Change directory to 15

Example 1 In serial mode, send a command to play 001.mp3.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3, 4); // RX, TX
const int BUSY = 5;
boolean cmd_stop =false;

void setup() {
 pinMode(BUSY, INPUT);
 mySerial.begin(9600);
}

void loop() {
  if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)) {
    Serial.write(0x01);
    cmd_stop=true;
  }
}
```

Code description:

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3, 4); // RX, TX
```

A software serial port named mySerial is established by using D3 as RX and D4 as TX.

```
void setup() {
 pinMode(BUSY, INPUT);
 mySerial.begin(9600);
}
```

BUSY (D5) is configured to be an input port. Then the serial port is initialized.

```
void loop() {
  if ((digitalRead(BUSY)  == HIGH) && (cmd_stop == false)) {
    mySerial.write(0x01);
    cmd_stop=true;
  }
}
```

When logic on BUSY pin becomes “High” and cmd_stop is false, a command 0x01 will be sent via serial port. Then cmd_stop is set to true. This will command the module to play the first file once.

Example 2 In serial mode, send a command to play 002.mp3 in folder 03.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(3, 4); // RX, TX

const int BUSY = 5;
boolean cmd_stop =false ;

void setup() {
pinMode(BUSY, INPUT);
mySerial.begin(9600);
}

void loop() {
   if ((digitalRead(BUSY) == HIGH) && (cmd_stop == false)) {
         mySerial.write(0xF3);
         delay(200);
          while(digitalRead(BUSY) == HIGH) {
           mySerial.write (0x02);
           cmd_stop=true;
        }
    }
}
```

Code description:

```
void loop() {
   if ((digitalRead(BUSY)  == HIGH) && (cmd_stop == false)) {
         mySerial.write(0xF3);
         delay(200);
          while(digitalRead(BUSY) == HIGH) {
           mySerial.write (0x02);
           cmd_stop=true;
        }
    }
}
```

In loop, when D5 (BUSY) becomes “High” and cmd_stop is false, a command 0xF3 will be sent via serial port. This will change directory to 03. Then wait until BUSY becomes “High” again and a command 0x02 is sent to play 002.mp3.